PASSED

732A38: Computer lab 1
Computational statistics

Martina Sandberg and Caroline Svahn

February 4, 2016

1 Be careful with '=='

A pupil of a school is bad in arithmetic but good in programming. He writes a program to check if 1/3-1/4==1/12.

1.1

Check the result of this program. Comment why this happened.

When running this program we get the "Teacher lied" message. The reason for this is that the computer uses floating-point numbers and the real number 1/3 does not have an exact representation by a floating-point number because it has no finite representation in base 10 (or base 2). The so called "floats" are representations of the real numbers, however, do not entirely correspond to the real numbers. This occurs when there is no finite representation of a number with base 10, or, more accurately, when the denominator of the fraction of a real number cannot be factorized in only primes of the base. For our 10-base system, these primes are 2 and 5, whilst for computers, using a 2-base system, the only prime is 2. When the correct factorization cannot be done for a rational number, the computer instead uses the closest number to the number in question. This means the obtained number is not entirely accurate, causing this type of program to fail. So the rounding of the number becomes a problem in this program.

1.2

Specify how the program can be modified to give a correct result.

Since "==" means exactly equal to, and as mentioned the computer cannot represent 1/3 in its exact form, we should use another way of comparing the numbers. One can use the command `all.equal()` instead in which a tolerance level can be set, allowing some deviance between the values in question.

2 Derivative

A widely known way to compute the derivative of function $f(x)$ in point x is to use

$$f'^{(x)} = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

2.1

Write your own function computing the derivative of function f(x)=x in this way. Take $\varepsilon = 10^{-15}$.

2.2

Compute your derivative function at point x=100000.

2.3

What is the value you obtained? What is the real value of the derivative? Explain the reason behind the discovered difference.

The obtained number is 0. However, it should return 1:

$$f'^{(x)} = \frac{100000 + 10^{-15} - 100000}{10^{-15}} = \frac{10^{-15}}{10^{-15}} = 1.$$

The reason why the function does not yield the correct value is because the magnitude of the two variables are too varying. When adding the very small number 10^{-15} to the significantly larger 100000, the precision of the smaller number is lost in the great magnitude of x. Thus, what is actually obtained from the function is:

$$f'^{(x)} = \frac{100000 + 10^{-15} - 100000}{10^{-15}} = \frac{100000 - 100000}{10^{-15}} = \frac{0}{10^{-15}} = 0.$$

To prevent this situation, one approach is to sort the numbers by magnitude and adding the numbers closest to magnitude in order to keep the precision. This approach would give:

$$f'^{(x)} = \frac{100000 + 10^{-15} - 100000}{10^{-15}} = \frac{100000 - 100000 + 10^{-15}}{10^{-15}} = \frac{10^{-15}}{10^{-15}} = 1.$$

3 Variance

A known formula for estimating the variance is

$$Var(x) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

3.1

Write your own function myvar estimating variance in this way.

3.2

Generate vector $x = (x_1 \dots x_{10000})$ *with 10000 random numbers, normally distributed with mean* 10^8 *and variance 1.*

3.3

For each subset $X_i = \{x_1 \dots x_i\}$, $i = 1, \dots, 1000$ *compute difference* $Y_i = myvar(X_i) - var(X_i)$, *where* $var(X_i)$ *is a standard variance estimation function in R. Plot the dependence* Y_i *on i. Draw necessary conclusions. How well does your function work? What is the reason behind such behavior?*

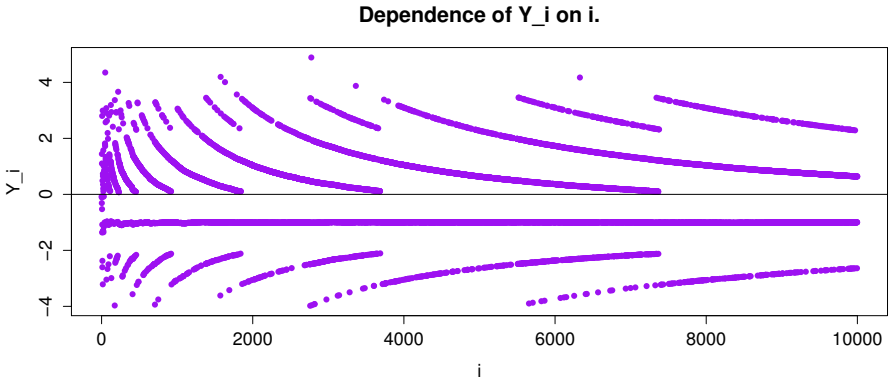

Figure 1: Dependece of $Y_i = myvar(X_i) - var(X_i)$ *on* i.

If we see $var(X_i)$ as the actual variance we want, Y_i ought to be 0. We can see in figure 1 that this never occurs. We can also see that the difference is often around -1, that is, our function `myvar()` is often one unit smaller than the actual variance. We can see some overlapping interval patterns which starts far from zero but then approaches it. These intervals also get wider for larger subsets of x. Our function returns zero variance when it should return a variance of 1. The reason for this could be catastrophic cancellation, i.e the rounding of the numbers makes the difference between them "disappear" or get miscalculated. The rounding causing the cancellation is due to the limited precision of the floats, mentioned earlier. This applies to values which have approximately equal magnitude.

4 Linear Algebra

The Excel file `tecator.xls` contains the results of a study aimed to investigate whether a near- infrared absorbance spectrum and the levels of moisture and fat can be used to predict the protein content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry. The worksheet you need to use is "data". It contains data from 215 samples of finely chopped meat. The aim is to fit a linear regression model that could predict protein content.

4.1

Import the data set to R.

4.2

Optimal regression coefficients can be found by solving a system of the type $A\beta = b$ where $A = X^T X$ and $b = X^T Y$. Compute A and b for the given data set.

4.3

Try to solve $A\beta = b$ with default solver `solve()`. What kind of result did you get? How can this result be explained?

When trying to solve the system we get an error. The function found the system to be singular, so the matrix A seems to have (or have nearly) linear dependencies, i.e strongly correlated variables. This results in no existing inverse of **A**. Checking the reciprocal condition number, which is rather close to zero, one can suspect that this result is due to an ill-conditioned matrix **A** (formula for reciprocal condition number is given in next subsection).

4.4

Check the condition number of the matrix A and consider how it is related to your conclusion in step 3.

The condition number is defined as

$$\kappa(\mathbf{A}) = ||\mathbf{A}|| \times ||\mathbf{A}^{-1}||$$

Using the `kappa()` function to find the condition number, the worst case numerical error, of the matrix **A** gives 1157834000000000, which is very high. This indicates the matrix to be of bad condition, which is probably why the `solve()` function found the system to be singular. Since we do not have an inverse in this case the computer uses a pseudo inverse for the computation of the condition number.

The reciprocal condition number is $\frac{1}{\kappa(\mathbf{A})}$, which is why a value close to zero is an indication of ill-conditioned data.

4.5

Scale the data set and repeat steps 2-4. How the result has changed and why?

This time we can solve the system. The conditional number in this case is 490471520662. This is much smaller than the one obtained in 4.4, which implies that this result is more reliable than the previous one. Singularity of a matrix may occur when variables or rows in the matrix are correlated (among other reasons), which is likely to be the case here since scaling helped.

Appendix

Code

```
#1


#1.1

x1<-1/3;
x2<-1/4;

if (x1-x2==1/12){
  print("Teacher said true")
} else{
  print("Teacher lied")}

#1.2

if (isTRUE(all.equal(x1-x2,1/12))){
  print("Teacher said true")
} else{
  print("Teacher lied")}

#2.1
derive<-function(x) {
  out<-((x+(10^(-15)))-x)/(10^(-15))
  return(out)
}

#2.2
derive(100000)

#2.3
#0, should be 1

#3.1
myvar<-function(x) {
  n<-length(x)
  df<-1/(n-1)

  s<-sum(x^2)
  s2<-sum(x)^2

  var<-df*(s-(1/n)*(s2))
  return(var)
}

#3.2
x<-rnorm(10000, mean=10^8)
```

```
#3.3

y<-numeric()

for (i in 1:length(x)) {
  subset<-x[1:i]
  y[i]<-myvar(subset)-var(subset)
}

plot(y, xlab="i", ylab="Difference", main="Dependence (Y on i)",
        col="seagreen", type="l")

#4.1
library(xlsx)
tecator<-read.xlsx("D:/Dokument/732A38/Lab1/tecator.xls",
        sheetName="data")

#4.2
Y<-tecator[,"Protein"]
X<-as.matrix(tecator[,c(-1,-103)])

A<-t(X)%*%X
b<-t(X)%*%Y

#4.3
solve(A,b)
#system is computationally singular:
        reciprocal condition number=7.13971e-17

#4.4
kappa(A)
#1.157834e+15

#4.5
sX<-scale(X)
sY<-scale(Y)

sA<-t(sX)%*%sX
sb<-t(sX)%*%sY

#4.3
solve(sA,sb)

#4.4
kappa(sA)
```